To create all the different Binary Search Trees (BSTs) that can represent the sequence of keys **"best of it the time was"**, we need to consider each word as a unique key and construct all possible valid BSTs. This is an interesting problem that requires careful consideration of the BST property and permutations.

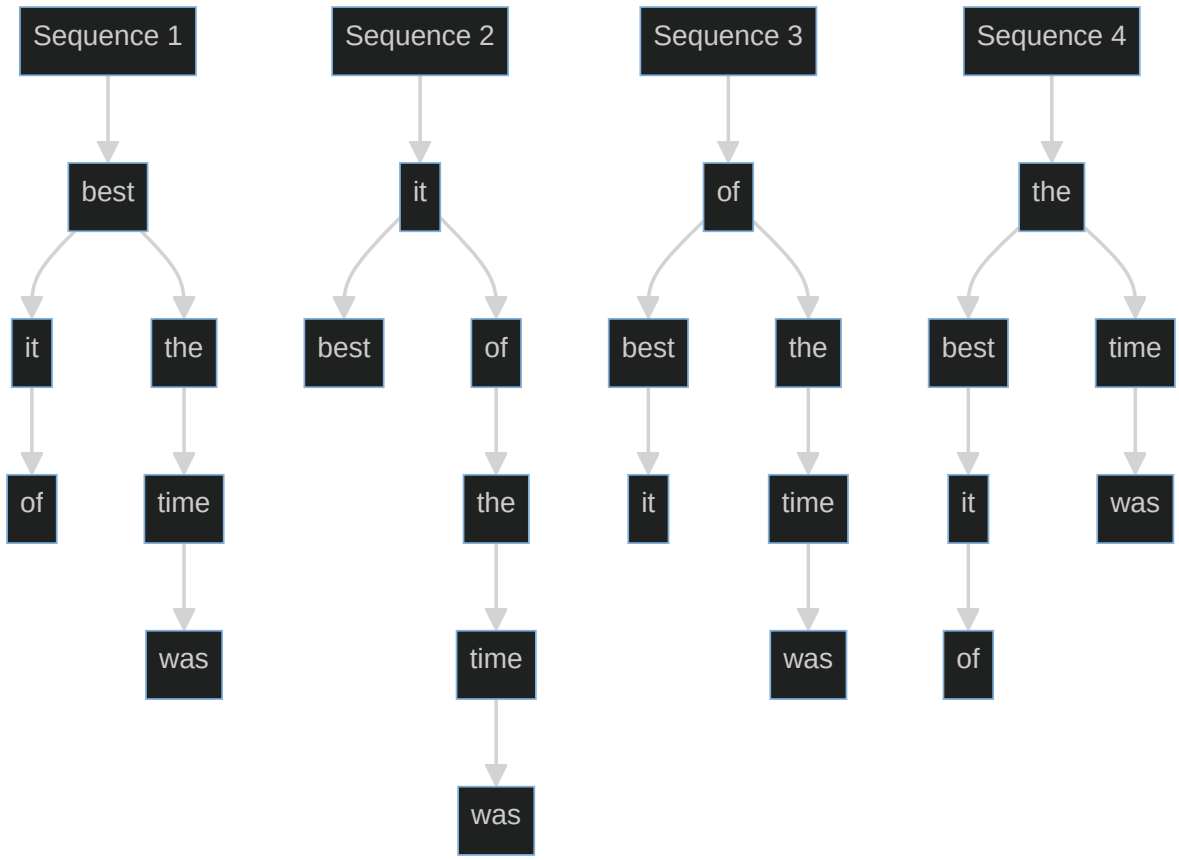
Let's approach this step-by-step:

1. First, we need to identify the unique keys in alphabetical order: best, it, of, the, time, was
2. Now, we'll construct all possible BSTs using these keys. The number of possible BSTs can be quite large, so I'll draw a few examples to illustrate the process.

In [1]:
```
from IPython.display import SVG

def show_svg(string):
    return SVG(string)
show_svg("t.svg")
```

Out[1]:



Each of these trees satisfies the BST property, where for each node:

- All keys in the left subtree are less than the node's key
- All keys in the right subtree are greater than the node's key

It's important to note that these are just a few examples out of many possible valid BSTs. The total number of possible BSTs for n unique keys is given by the nth Catalan number. In this case, with 6 unique keys, there are actually 132 different possible BSTs!

Some observations about the examples:

1. Sequence 1 has "best" as the root, with most words in the right subtree.
2. Sequence 2 has "it" as the root, creating a more balanced tree.
3. Sequence 3 uses "of" as the root, splitting the words more evenly.
4. Sequence 4 has "the" as the root, with most words in the left subtree.

Each of these trees would result in a different in-order traversal, but they all represent valid BSTs for the given set of keys.